Chapter preprint of the second edition of the Handbook of Satisfiability (2021)
Armin Biere, Marijn Heule, Hans van Maaren and Toby Walsh (Eds.)
Published by IOS Press. The print edition can be ordered at
https://www.iospress.nl/book/handbook-of-satisfiability-2/
The published electronic chapters (Version of Record) are available at
https://ebooks.iospress.nl/ISBN/978-1-64368-161-0/

Chapter 23

MaxSAT, Hard and Soft Constraints

Chu Min Li and Felip Manyà

23.1. Introduction

MaxSAT is an optimization version of SAT which consists in finding an assignment that maximizes the number of satisfied clauses. It is an NP-hard problem that has shown a remarkable activity in the SAT community over the past few years, where some SAT solving techniques have been adapted to MaxSAT and incorporated into contemporary MaxSAT solvers. Examples of such SAT solving techniques include lazy data structures, variable selection heuristics, unsatisfiable core extraction, non-chronological backtracking and clause learning.

SAT solvers provide little information on unsatisfiable instances; they just report that no solution exists. However, assignments violating a minimum number of constraints, or satisfying all the hard constraints and as many soft constraints as possible, can be considered acceptable solutions in real-life scenarios. To cope with this limitation of SAT, MaxSAT, and in particular weighted MaxSAT and Partial MaxSAT, are becoming an alternative for naturally representing and efficiently solving over-constrained problems.

There are two approaches to solving MaxSAT: approximation and heuristic algorithms, which compute near-optimal solutions, and exact algorithms, which compute optimal solutions. Heuristic algorithms are fast and do not provide any guarantee about the quality of their solutions, while approximation algorithms are not so fast but provide a guarantee about the quality of their solutions.

Regarding exact algorithms, we distinguish between SAT-based algorithms and branch-and-bound algorithms. SAT-based algorithms solve MaxSAT by solving a sequence of SAT instances and are presented in a separate chapter of this handbook. The solving techniques that implement branch-and-bound algorithms are the main focus of this chapter, and are explained in detail in subsequent sections.

Nowadays, we count with remarkable results on theoretical, logical and algorithmic aspects of MaxSAT solving. Moreover, the existence of a MaxSAT evaluation, which is held annually since 2006, has been decisive for developing new MaxSAT technology and applications.

The structure of the chapter is as follows. In Section 23.2, we introduce background definitions. In Section 23.3, we present the branch and bound scheme and explain how this scheme can be improved with good quality lower bounds, clever variable selection heuristics and suitable data structures. In Section 23.4, we survey the complete logical calculus for MaxSAT defined so far, which are based on resolution and tableaux. In Section 23.5, we review the main MaxSAT approximation algorithms. In Section 23.6, we present the MaxSAT Evaluation. In Section 23.7, we review other contributions to MaxSAT that have appeared in the literature. In Section 23.8, we survey the works on MinSAT, which is the dual problem of MaxSAT. In Section 23.9, we give the conclusions and point out some future research directions.

23.2. Preliminaries

In propositional logic, a variable x_i may take values 0 (for false) or 1 (for true). A literal l_i is a variable x_i or its negation $\bar{x}_i$. The complementary of literal l_i is x_i if $l_i = \bar{x}_i$, and is $\bar{x}_i$ if $l_i = x_i$. A clause is a disjunction of literals, and a CNF formula is a multiset of clauses. In MaxSAT, we represent CNF formulas as multisets of clauses instead of sets of clauses because duplicated clauses cannot be collapsed into one clause. For instance, the multiset $\{x_1, \overline{x}_1, \overline{x}_1, x_1 \vee x_2, \overline{x}_2\}$, where a clause is repeated, has a minimum of two unsatisfied clauses.

A weighted clause is a pair (C_i, w_i), where C_i is a disjunction of literals and w_i, its weight, is a positive number, and a weighted CNF formula is a multiset of weighted clauses. The length of a (weighted) clause is the number of its literals. The size of (weighted) CNF formula ϕ, denoted by $|\phi|$, is the sum of the length of all its clauses.

An assignment of truth values to the propositional variables satisfies a literal x_i if x_i takes the value 1 and satisfies a literal $\bar{x}_i$ if x_i takes the value 0, satisfies a clause if it satisfies at least one literal of the clause, and satisfies a CNF formula if it satisfies all the clauses of the formula. An empty clause, denoted by $\square$, contains no literals and cannot be satisfied. An assignment for a CNF formula ϕ is complete if all the variables occurring in ϕ have been assigned; otherwise, it is partial.

The MaxSAT problem for a CNF formula ϕ is the problem of finding an assignment that maximizes the number of satisfied clauses. In this sequel we often use the term MaxSAT meaning MinUNSAT. This is because, with respect to exact computations, finding an assignment that minimizes the number of unsatisfied clauses is equivalent to finding an assignment that maximizes the number of satisfied clauses. Notice that an upper (lower) bound in MinUNSAT is greater (smaller) than or equal to the minimum number of clauses that can be unsatisfied by an interpretation. MaxSAT is called Max-kSAT when all the clauses have at most k literals per clause.

MaxSAT instances ϕ_1 and ϕ_2 are equivalent if ϕ_1 and ϕ_2 have the same number of unsatisfied clauses for every complete assignment of ϕ_1 and ϕ_2.

We will also consider three extensions of MaxSAT which are more well-suited for representing and solving over-constrained problems: Weighted MaxSAT, Partial MaxSAT, and weighted Partial MaxSAT.

The weighted MaxSAT problem for a weighted CNF formula ϕ is the problem of finding an assignment that maximizes the sum of weights of satisfied clauses (or equivalently, that minimizes the sum of weights of unsatisfied clauses).

The Partial MaxSAT problem for a CNF formula, in which some clauses are declared to be *relaxable* or *soft* and the rest are declared to be *non-relaxable* or *hard*, is the problem of finding an assignment that satisfies all the hard clauses and the maximum number of soft clauses.

The weighted Partial MaxSAT problem is the combination of Partial MaxSAT and weighted MaxSAT. In weighted Partial MaxSAT, soft clauses are weighted, and solving a weighted Partial MaxSAT instance amounts to find an assignment that satisfies all the hard clauses and maximizes the sum of weights of satisfied soft clauses (or equivalently, that minimizes the sum of weights of unsatisfied soft clauses).

Notice that MaxSAT could be defined as weighted MaxSAT restricted to formulas whose clauses have weight 1, and as Partial MaxSAT in the case that all the clauses are declared to be soft.

Finally, we introduce the integer linear programming (ILP) formulation of weighted MaxSAT, which is used to compute lower bounds and upper bounds. Let $\phi = \{(C_1, w_1), \ldots, (C_m, w_m)\}$ be a weighted MaxSAT instance over the propositional variables $x_1, \ldots, x_n$. With each propositional variable x_i, we associate a variable $y_i \in \{0, 1\}$ such that $y_i = 1$ if variable x_i is true and $y_i = 0$, otherwise. With each clause C_j, we associate a variable $z_j \in \{0, 1\}$ such that $z_j = 1$ if clause C_j is satisfied and $z_j = 0$, otherwise. Let I_j^+ be the set of indices of unnegated variables in clause C_j, and let I_j^- be the set of indices of negated variables in clause C_j. The ILP formulation of the weighted MaxSAT instance ϕ is defined as follows:

$$\max F(y, z) = \sum_{j=1}^{m} w_j z_j$$

subject to

$$\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq z_j \quad j = 1, \ldots, m$$

$$y_i \in \{0, 1\} \quad i = 1, \ldots, n$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, m$$

Assume now that, with each clause C_j, we associate a variable $z_j \in \{0, 1\}$ such that $z_j = 1$ if clause C_j is unsatisfied and $z_j = 0$, otherwise. Then, the ILP formulation of the minimization version of weighted MaxSAT (i.e.; weighted MinUNSAT) for the instance ϕ is defined as follows:

$$\min F(y, z) = \sum_{j=1}^{m} w_j z_j$$

subject to

$$\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) + z_j \geq 1 \quad j = 1, \ldots, m$$

$$y_i \in \{0,1\} \quad i = 1,\ldots,n$$

$$z_j \in \{0,1\} \quad j = 1,\ldots,m$$

The linear programming (LP) relaxation of both formulations is obtained by allowing the integer variables to take real values in $[0,1]$.

Ansótegui and Gabàs [AG13] reported an extensive empirical investigation that indicates that solving MaxSAT instances by translating them into ILP and applying a Mixed Integer Programming (MIP) solver is competitive on crafted instances.

Weighted Partial MinSAT for an instance ϕ is the problem of finding an assignment in which the sum of the weights of the satisfied soft clauses is minimal, and all the hard clauses are satisfied. Weighted MinSAT is Weighted Partial MinSAT when there are no hard clauses. Partial MinSAT is Weighted Partial MinSAT when all the soft clauses have the same weight. MinSAT is Partial MinSAT when there are no hard clauses.

23.3. Branch and Bound Algorithms

There are competitive exact MaxSAT solvers —as the ones developed by [AH14a, AMP03, AMP05, AMP08, HLO08, Kue10, LHdG08, LMP07, LS07, LSL08, PPC+08, RG07, SZ04, XZ04, XZ05, ZSM03]— that implement variants of the following branch and bound (BnB) scheme for solving the minimization version of MaxSAT: Given a MaxSAT instance ϕ, BnB explores the search tree that represents the space of all possible assignments for ϕ in a depth-first manner. At every node, BnB compares the upper bound (UB), which is the best solution found so far for a complete assignment, with the lower bound (LB), which is the sum of the number of clauses which are unsatisfied by the current partial assignment plus an underestimation of the number of clauses that will become unsatisfied if the current partial assignment is completed. If $LB \geq UB$, the algorithm prunes the subtree below the current node and backtracks chronologically to a higher level in the search tree. If $LB < UB$, the algorithm tries to find a better solution by extending the current partial assignment by instantiating one more variable. The optimal number of unsatisfied clauses in the input MaxSAT instance is the value that UB takes after exploring the entire search tree.

Figure 23.1 shows the pseudo-code of a basic solver for MaxSAT. We use the following notation:

- *simplifyFormula*(ϕ) is a procedure that transforms ϕ into an equivalent and simpler instance by applying inference rules.
- #*emptyClauses*(ϕ) is a function that returns the number of empty clauses in ϕ.
- LB is a lower bound of the minimum number of unsatisfied clauses in ϕ if the current partial assignment is extended to a complete assignment. We assume that its initial value is 0.
- *underestimation*(ϕ) is a function that returns an underestimation of the minimum number of non-empty clauses in ϕ that will become unsatisfied if the current partial assignment is extended to a complete assignment.

Require: $MaxSAT(\phi, UB)$: A MaxSAT instance ϕ and an upper bound UB
1: $\phi \leftarrow simplifyFormula(\phi)$;
2: **if** $\phi = \emptyset$ or ϕ only contains empty clauses **then**
3: return $\#emptyClauses(\phi)$;
4: **end if**
5: $LB \leftarrow \#emptyClauses(\phi) + underestimation(\phi)$;
6: **if** $LB \geq UB$ **then**
7: return UB;
8: **end if**
9: $x \leftarrow selectVariable(\phi)$;
10: $UB \leftarrow \min(UB, MaxSAT(\phi_{\bar{x}}, UB))$;
11: return $\min(UB, MaxSAT(\phi_x, UB))$;
Ensure: The minimal number of unsatisfied clauses in ϕ

Figure 23.1. A basic branch and bound algorithm for MaxSAT

- UB is an upper bound of the number of unsatisfied clauses in an optimal solution. An elementary initial value for UB is the total number of clauses in the input formula, or the number of clauses which are unsatisfied by an arbitrary interpretation. Another alternative is to solve the LP relaxation of the ILP formulation of the input instance and take as upper bound the number of unsatisfied clauses in the interpretation obtained by rounding variable y_i, for $1 \leq i \leq n$, to an integer solution in a randomized way by interpreting the values of $y_i \in [0, 1]$ as probabilities (set propositional variable x_i to true with probability y_i, and set propositional variable x_i to false with probability $1 - y_i$). Nevertheless, most of the solvers take as initial upper bound the number of unsatisfied clauses that can be detected by executing the input formula in a local search solver during a short period of time.
- $selectVariable(\phi)$ is a function that returns a variable of ϕ following an heuristic.
- ϕ_x ($\phi_{\bar{x}}$) is the formula obtained by setting the variable x to true (false); i.e., by applying the one-literal rule to ϕ using the literal x ($\bar{x}$).

Modern MaxSAT solvers implement the basic algorithm augmented with powerful inference techniques, good quality lower bounds, clever variable selection heuristics, and efficient data structures. Partial MaxSAT solvers are also augmented with learning of hard clauses, and non-chronological backtracking.

23.3.1. Improving the Lower Bound with Underestimations

The simplest method to compute a lower bound, when solving the minimization version of MaxSAT, consists in just counting the number of clauses which are unsatisfied by the current partial assignment [BF99]. One step forward is to incorporate an underestimation of the number of clauses that will become unsatisfied if the current partial assignment is extended to a complete assignment. The most basic method was defined by Wallace and Freuder [WF96]:

$$\texttt{LB}(\phi) = \#emptyClauses(\phi) + \sum_{x \text{ occurs in } \phi} min(ic(x), ic(\bar{x})),$$

where ϕ is the CNF formula associated with the current partial assignment, and $ic(x)$ ($ic(\bar{x})$) —inconsistency count of x ($\bar{x}$)— is the number of unit clauses of ϕ that contain $\bar{x}$ (x). In other words, that underestimation is the number of disjoint inconsistent subformulas in ϕ formed by a unit clause with a literal l and a unit clause with the complementary of l.

Lower bounds dealing with longer clauses include the star rule and UP. In the star rule [SZ04, AMP04], the underestimation of the lower bound is the number of disjoint inconsistent subformulas of the form $\{l_1, \ldots, l_k, \bar{l}_1 \vee \cdots \vee \bar{l}_k\}$. When $k = 1$, the star rule is equivalent to the inconsistency counts of Wallace and Freuder.

In UP [LMP05], the underestimation of the lower bound is the number of disjoint inconsistent subformulas that can be detected with unit propagation. UP works as follows: It applies unit propagation until a contradiction is derived. Then, UP identifies, by inspecting the implication graph, a subset of clauses from which a unit refutation can be constructed, and tries to identify new contradictions from the remaining clauses. The order in which unit clauses are propagated has a clear impact on the quality of the lower bound [LMP06].

UP can be enhanced with failed literal detection as follows: Given a MaxSAT instance ϕ and a variable x occurring positively and negatively in ϕ, UP is applied to both $\phi \wedge \{x\}$ and $\phi \wedge \{\bar{x}\}$. If UP derives a contradiction from $\phi \wedge \{x\}$ and another contradiction from $\phi \wedge \{\bar{x}\}$, then the union of the two inconsistent subsets identified by UP, once we have removed the unit clauses x and $\bar{x}$, is an inconsistent subset of ϕ. UP enhanced with failed literal detection does not need the occurrence of unit clauses in the input formula for deriving a contradiction. While UP only identifies unit refutations, UP enhanced with failed literal detection identifies non-unit refutations too. Since applying failed literal detection to every variable is time consuming, it is applied to a reduced number of variables in practice [LMP06].

MaxSAT solvers like ahmaxsat [AH14a], MaxSatz [LMP07], and Mini-MaxSat [HLO08] apply either UP or UP enhanced with failed literal detection. Nowadays, UP-based lower bounds are the prevailing approach to computing understimations in branch-and-bound MaxSAT solvers. This technique has also been applied to solve the maximum clique problem [JLM17, LFX13, LJM17, LJX15, LQ10].

Darras et al. [DDDL07] developed a version of UP in which the computation of the lower bound is made more incremental by saving some of the small size disjoint inconsistent subformulas detected by UP. They avoid to redetect the saved inconsistencies if they remain in subsequent nodes of the proof tree, and are able to solve some types of instances faster. Lin et al. [LSL08] defined an improved version of UP that, besides being incremental, guarantees that the lower bound computed at a node of the search tree is not smaller than the lower bound computed at the parent of that node. Abramé and Habet [AH14c] proposed an improved implementation of UP in ahmaxsat that undoes propagations in a non-chronological order and can produce smaller inconsistent subsets. Shen and Zhang [SZ04] defined a lower bound, called LB4, which is similar to UP but restricted to Max-2SAT instances and using a static variable ordering.

A variant of UP enhanced with failed literal detection was implemented in the solver akmaxsat [Kue10]. It can be the case that UP derives a contradiction

from $\phi \wedge \{l\}$ but not from $\phi \wedge \{\bar{l}\}$. In fact, this shows that $\bar{l}$ follows from ϕ. If ϕ' is the result of applying UP to $\phi \wedge \{\bar{l}\}$, then the algorithms tries to find another failed literal l' in ϕ'. If UP derives a contradiction from both $\phi \wedge \{l'\}$ and $\phi \wedge \{\bar{l}'\}$, the algorithm stops and identifies an inconsistent subset. If UP derives a contradiction from $\phi \wedge \{l'\}$ but not from $\phi \wedge \{\bar{l}'\}$, the same process is repeated on the formula resulting of applying UP to $\phi \wedge \{\bar{l}'\}$ until an inconsistent subset is detected or no more failed literals can be found.

Another approach to computing underestimation is based on first reducing the MaxSAT instance to be solved to an instance of another problem, and then solve a relaxation of the obtained instance. For example, the solvers Clone [PD07, PPC+08] and SR(w) [RG07], reduce MaxSAT to the minimum cardinality problem. Since the minimum cardinality problem is NP-hard for a CNF formula ϕ and can be solved in time linear in the size of a deterministic decomposable negation normal form (d-DNNF) compilation of ϕ, Clone and SR(w) solve the minimum cardinality problem of a d-DNNF compilation of a relaxation of ϕ. The worst-case complexity of a d-DNNF compilation of ϕ is exponential in the treewidth of its constraint graph, and Clone and SR(w) obtain a relaxation of ϕ with bounded treewidth by renaming different occurrences of some variables.

Xing and Zhang [XZ05] reduce the MaxSAT instance to the ILP formulation of the minimization version of MaxSAT (c.f. Section 23.2), and then solve the LP relaxation. An optimal solution of the LP relaxation provides an underestimation of the lower bound because the LP relaxation is less restricted than the ILP formulation. In practice, they apply that lower bound computation method only to nodes containing unit clauses. If each clause in the MaxSAT instance has more than one literal, then $y_i = \frac{1}{2}$ for all $1 \leq i \leq n$ and $z_j = 0$ for all $1 \leq j \leq m$ is an optimal solution of the LP relaxation. In this case, the underestimation is 0. Nevertheless, LP relaxations do not seem to be as competitive as the rest of approaches.

23.3.2. Improving the Lower Bound with Inference

Another approach to improve the quality of the lower bound consists in applying inference rules that transform a MaxSAT instance ϕ into an *equivalent* but simpler MaxSAT instance ϕ'. In the best case, inference rules produce new empty clauses in ϕ' that allow to increment the lower bound. In contrast with the empty clauses derived when computing underestimations, the empty clauses derived with inference rules do not have to be recomputed at every node of the current subtree so that the lower bound computation is more incremental.

A MaxSAT inference rule is *sound* if it transforms an instance ϕ into an equivalent instance ϕ'. It is not sufficient to preserve satisfiability as in SAT, ϕ and ϕ' must have the same number of unsatisfied clauses for every possible assignment. Unfortunately, unit propagation, which is the most powerful inference technique applied in DPLL-style SAT solvers, is unsound for MaxSAT as the next example shows: The set of clauses $\{x_1, \overline{x}_1 \vee x_2, \overline{x}_1 \vee \overline{x}_2, \overline{x}_1 \vee x_3, \overline{x}_1 \vee \overline{x}_3\}$ has a minimum of one unsatisfied clause (setting x_1 to false), but two empty clauses are derived by applying unit propagation.

MaxSAT inference rules are also called *transformation rules* in the literature

because the premises of the rule are replaced with the conclusion when a rule is applied. If the conclusion is added to the premises as in SAT, the number of clauses which are unsatisfied by an assignment might increase.

The amount of inference enforced by existing BnB MaxSAT solvers at each node of the proof tree is poor compared with the inference enforced by DPLL-style SAT solvers. The simplest inference enforced, when branching on a literal l, consists in applying the one-literal rule: The clauses containing l are removed from the instance and the occurrences of $\bar{l}$ are removed from the clauses in which $\bar{l}$ appears, but the existing unit clauses and the new unit clauses derived as a consequence of removing the occurrences of $\bar{l}$ are not propagated as in unit propagation. That inference is typically enhanced with the MaxSAT inference rules described in the rest of this section.

First, we present simple inference rules that have proved to be useful in a number of solvers [AMP03, AMP05, BF99, SZ04, XZ05], and then some more sophisticated inferences rules which are implemented in solvers like ahmaxsat [AH14a], akmaxsat [Kue10], MaxSatz [LMP07], and MiniMaxSat [HLO08]. Some simple inference rules are:

- The pure literal rule [BF99]: If a literal only appears with either positive or negative polarity in a MaxSAT instance, all the clauses containing that literal are removed.
- The dominating unit clause rule [NR00]: If the number of clauses (of any length) in which a literal l appears is not greater than the number of unit clauses in which $\bar{l}$ appears, all the clauses containing l and all the occurrences of $\bar{l}$ are removed.
- The complementary unit clause rule [NR00]: If a MaxSAT instance contains a unit clause with the literal l and a unit clause with the literal $\bar{l}$, these two clauses are replaced with one empty clause.
- The almost common clause rule [BR99]: If a MaxSAT instance contains a clause $x \vee D$ and a clause $\overline{x} \vee D$, where D is a disjunction of literals, then both clauses are replaced with D. In practice, this rule is applied when D contains at most one literal.

The resolution rule applied in SAT (i.e., derive $D \vee D'$ from $x \vee D$ and $\overline{x} \vee D'$) preserves satisfiability but not equivalence, and therefore cannot be applied to MaxSAT instances, except for some particular cases like the almost common clause rule. We refer the reader to Section 23.4 for a complete resolution calculus for MaxSAT, and devote the rest of this section to present some sound MaxSAT resolution rules that can be applied in polynomial time.

We start by presenting the *star rule*: If $\phi_1 = \{l_1,\ \bar{l}_1 \vee \bar{l}_2,\ l_2\} \cup \phi'$, then $\phi_2 = \{\square,\ l_1 \vee l_2\} \cup \phi'$ is equivalent to ϕ_1. This rule, which can be seen as the inference counterpart of the underestimation of the same name, can also be presented as follows:

$$\left\{ \begin{array}{c} l_1 \\ \bar{l}_1 \vee \bar{l}_2 \\ l_2 \end{array} \right\} \Longrightarrow \left\{ \begin{array}{c} \square \\ l_1 \vee l_2 \end{array} \right\} \tag{23.1}$$

Notice that the rule detects a contradiction from $l_1, \bar{l}_1 \vee \bar{l}_2, l_2$ and, therefore,

replaces these clauses with an empty clause. In addition, the rule adds the clause $l_1 \vee l_2$ to ensure the equivalence between ϕ_1 and ϕ_2. For any assignment containing either $l_1 = 0, l_2 = 1$, or $l_1 = 1, l_2 = 0$, or $l_1 = 1, l_2 = 1$, the number of unsatisfied clauses in $\{l_1, \bar{l}_1 \vee \bar{l}_2, l_2\}$ is 1, but for any assignment containing $l_1 = 0, l_2 = 0$, the number of unsatisfied clauses is 2. Notice that even when any assignment containing $l_1 = 0, l_2 = 0$ is not the best assignment for the subset $\{l_1, \bar{l}_1 \vee \bar{l}_2, l_2\}$, it can be the best for the whole formula. By adding $l_1 \vee l_2$, the rule ensures that the number of unsatisfied clauses in ϕ_1 and ϕ_2 is also the same when $l_1 = 0, l_2 = 0$.

This rule can be generalized in such a way that it captures unit resolution refutations in which clauses and resolvents are used exactly once:

$$\left\{\begin{array}{c} l_1 \\ \bar{l}_1 \vee l_2 \\ \bar{l}_2 \vee l_3 \\ \cdots \\ \bar{l}_k \vee l_{k+1} \\ \bar{l}_{k+1} \end{array}\right\} \Longrightarrow \left\{\begin{array}{c} \square \\ l_1 \vee \bar{l}_2 \\ l_2 \vee \bar{l}_3 \\ \cdots \\ l_k \vee \bar{l}_{k+1} \end{array}\right\} \tag{23.2}$$

The last two rules consume two unit clauses for deriving one contradiction. Next, we define two inference rules that capture unit resolution refutations in which (i) exactly one unit clause is consumed, and (ii) the unit clause is used twice in the derivation of the empty clause. The second rule is a combination of the first rule with a linear derivation.

$$\left\{\begin{array}{c} l_1 \\ \bar{l}_1 \vee l_2 \\ \bar{l}_1 \vee l_3 \\ \bar{l}_2 \vee \bar{l}_3 \end{array}\right\} \Longrightarrow \left\{\begin{array}{c} \square \\ l_1 \vee \bar{l}_2 \vee \bar{l}_3 \\ \bar{l}_1 \vee l_2 \vee l_3 \end{array}\right\} \tag{23.3}$$

$$\left\{\begin{array}{c} l_1 \\ \bar{l}_1 \vee l_2 \\ \bar{l}_2 \vee l_3 \\ \cdots \\ \bar{l}_k \vee l_{k+1} \\ \bar{l}_{k+1} \vee l_{k+2} \\ \bar{l}_{k+1} \vee l_{k+3} \\ \bar{l}_{k+2} \vee \bar{l}_{k+3} \end{array}\right\} \Longrightarrow \left\{\begin{array}{c} \square \\ l_1 \vee \bar{l}_2 \\ l_2 \vee \bar{l}_3 \\ \cdots \\ l_k \vee \bar{l}_{k+1} \\ l_{k+1} \vee \bar{l}_{k+2} \vee \bar{l}_{k+3} \\ \bar{l}_{k+1} \vee l_{k+2} \vee l_{k+3} \end{array}\right\} \tag{23.4}$$

MaxSatz implements the almost common clause rule, Rule 23.1, Rule 23.2, Rule 23.3 and Rule 23.4. Some of these rules are also applied in the solvers ahmaxsat and akmaxsat.

Independently and in parallel to the definition of the rules of MaxSatz, similar inference rules were defined for weighted MaxSAT by Heras and Larrosa [LH05, HL06], and were implemented in Max-DPLL [LHdG08]. These rules were inspired by the soft local consistency properties defined in the constraint programming community [dGLMS03]. The rules implemented in Max-DPLL are the almost common clause rule, chain resolution and cycle resolution. Chain resolution,

which allows one to derive a new empty clause, is defined as follows:

$$\left\{\begin{array}{l}(l_1, w_1),\\ (\bar{l}_i \vee l_{i+1}, w_{i+1})_{1\leq i<k},\\ (\bar{l}_k, w_{k+1})\end{array}\right\} \Longrightarrow \left\{\begin{array}{l}(l_i, m_i - m_{i+1})_{1\leq i\leq k},\\ (\bar{l}_i \vee l_{i+1}, w_{i+1} - m_{i+1})_{1\leq i<k},\\ (l_i \vee \bar{l}_{i+1}, m_{i+1})_{1\leq i<k},\\ (\bar{l}_k, w_{k+1} - m_{k+1}),\\ (\square, m_{k+1})\end{array}\right\} \tag{23.5}$$

where w_i, $1 \leq i \leq k+1$, is the weight of the corresponding clause, and $m_i = \min(w_1, w_2, \ldots, w_i)$. Chain resolution is equivalent to Rule 23.2 if it is applied to unweighted MaxSAT.

Cycle resolution, which allows one to derive a new unit clause and whose application is restricted to $k = 3$ in Max-DPLL, is defined as follows:

$$\left\{\begin{array}{l}(\bar{l}_i \vee l_{i+1}, w_i)_{1\leq i<k},\\ (\bar{l}_1 \vee \bar{l}_k, w_k)\end{array}\right\} \Longrightarrow \left\{\begin{array}{l}(\bar{l}_1 \vee l_i, m_{i-1} - m_i)_{2\leq i\leq k},\\ (\bar{l}_i \vee l_{i+1}, w_i - m_i)_{2\leq i<k},\\ (\bar{l}_1 \vee l_i \vee \bar{l}_{i+1}, m_i)_{2\leq i<k},\\ (l_1 \vee \bar{l}_i \vee l_{i+1}, m_i)_{2\leq i<k},\\ (\bar{l}_1 \vee \bar{l}_k, w_k - m_k),\\ (\bar{l}_1, m_k)\end{array}\right\} \tag{23.6}$$

An analysis of the impact of cycle resolution on the performance of MaxSAT solvers can be found in [LMMP08, LMMP09, LMMP10].

A more general inference scheme is implemented in MiniMaxSat [HLO07, HLO08]. It detects a contradiction with unit propagation and identifies an unsatisfiable subset. Then, it creates a refutation for that unsatisfiable subset and applies the MaxSAT resolution rule defined in Section 23.4 if the size of the largest resolvent in the refutation is smaller than 4.

The lower bound computation methods based on unit propagation represent the different derivations of unit clauses in a graph, called implication graph [LMP07]. Looking at that graph, solvers identify the clauses which are involved in the derivation of a contradiction. In contemporary MaxSAT solvers, this graph is also used to decide whether the clauses involved in a contradiction match with the premises of the above mentioned inference rule.

Abramé and Habet [AH15b] showed that in some cases it is better not to apply MaxSAT resolution to certain inconsistent subsets of clauses because the transformations derived do not allow to detect further inconsistent subsets. They also showed that in other cases further inconsistent subsets can be detected if MaxSAT resolution is applied locally [AH14b].

23.3.3. Variable Selection Heuristics

Most of the exact MaxSAT solvers incorporate variable selection heuristics that take into account the number of literal occurrences in such a way that each occurrence has an associated weight that depends on the length of the clause that contains the literal. MaxSAT heuristics give priority to literals occurring in binary clauses instead of literals occurring in unit clauses as SAT heuristics do.

Let us see as an example the variable selection heuristic of MaxSatz [LMP07]: Let $neg1(x)$ ($pos1(x)$) be the number of unit clauses in which x is negative (positive), $neg2(x)$ ($pos2(x)$) be the number of binary clauses in which x is negative (positive), and let $neg3(x)$ ($pos3(x)$) be the number of clauses containing three or more literals in which x is negative (positive). MaxSatz selects the variable x such that $(neg1(x)+4*neg2(x)+neg3(x))*(pos1(x)+4*pos2(x)+pos3(x))$ is the largest. Once a variable x is selected, MaxSatz applies the following value selection heuristic: If $neg1(x)+4*neg2(x)+neg3(x) < pos1(x)+4*pos2(x)+pos3(x)$, set x to true; otherwise, set x to false. The solver ahmaxsat [AH14a] implements a variant of this variable selection heuristic.

Earlier MaxSAT solvers (AMP [AMP03], Lazy [AMP05], MaxSolver [XZ05], Max-DPLL [LHdG08], ...) incorporate variants of the two-sided Jeroslow rule that give priority to variables occurring often in binary clauses. MaxSolver changes the weights as the search proceeds.

23.3.4. Data Structures

Data structures for SAT have been naturally adapted to MaxSAT. We can divide the solvers into two classes: solvers like ahmaxsat, akmaxsat and MaxSatz representing formulas with adjacency lists, and solvers like Lazy and MiniMaxSat which use data structures with watched literals. Lazy data structures are particularly good when there is a big number of clauses; for example, in Partial MaxSAT solvers with clause learning.

23.4. Complete Inference in MaxSAT

23.4.1. MaxSAT Resolution

A natural extension to MaxSAT of the resolution rule applied in SAT was defined by Larrosa and Heras [LH05]:

$$\begin{array}{c} x \vee A \\ \overline{x} \vee B \\ \hline\hline A \vee B \\ x \vee A \vee \overline{B} \\ \overline{x} \vee \overline{A} \vee B \end{array}$$

However, two of the conclusions of this rule are not in clausal form, and the application of distributivity results into an unsound rule: Assume that $A = a_1$, and $B = b_1 \vee b_2$. Then, the interpretation that assigns false to x and a_1, and true to b_1 and b_2 unsatisfies one clause from the premises $(x \vee a_1)$ and unsatisfies two clauses from the conclusion of the rule $(x \vee a_1 \vee \overline{b}_1, x \vee a_1 \vee \overline{b}_2)$. Since the rule does not preserve the number of unsatisfied clauses, it is unsound.

Independently and in parallel, Bonet et al. [BLM06, BLM07], and Heras and Larrosa [HL06] defined a sound version of the previous rule with the conclusions in clausal form:

$$
\begin{array}{l}
x \vee a_1 \vee \cdots \vee a_s \\
\overline{x} \vee b_1 \vee \cdots \vee b_t \\
\hline\hline
a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_t \\
x \vee a_1 \vee \cdots \vee a_s \vee \overline{b_1} \\
x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \overline{b_2} \\
\cdots \\
x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_{t-1} \vee \overline{b_t} \\
\overline{x} \vee b_1 \vee \cdots \vee b_t \vee \overline{a_1} \\
\overline{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \overline{a_2} \\
\cdots \\
\overline{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \cdots \vee a_{s-1} \vee \overline{a_s}
\end{array}
$$

This inference rule concludes, apart from the conclusion where a variable has been cut, some additional clauses that contain one of the premises as subclause. We say that the rule *cuts* the variable x. The tautologies concluded by the rule are removed, and the repeated literals in a clause are collapsed into one.

Notice that an instance of MaxSAT resolution not only depends on the two premises and the cut variable (like in resolution), but also on the order of the literals in the premises. Notice also that, like in resolution, this rule concludes a new clause not containing the variable x, except when this clause is a tautology.

Bonet et al. [BLM06, BLM07] proved the completeness of MaxSAT resolution: By *saturating* successively w.r.t. all the variables, one derives as many empty clauses as the minimum number of unsatisfied clauses in the MaxSAT input instance. Saturating w.r.t. a variable amounts to apply the MaxSAT resolution rule to clauses containing that variable until every possible application of the inference rule only introduces clauses containing that variable (since tautologies are eliminated). Once a MaxSAT instance is saturated w.r.t. a variable, all the clauses containing that variable are not considered to saturate w.r.t. another variable. We refer to [BLM07] for further technical details and for the weighted version of the rule.

We consider the multiset of clauses $\phi = \{\overline{x}_1, x_1 \vee x_2, x_1 \vee x_3, \overline{x}_3\}$ to illustrate how a variable saturation algorithm works. We start by considering variable x_1 and resolve the first two clauses, obtaining $\{x_2, \overline{x}_1 \vee \overline{x}_2, x_1 \vee x_3, \overline{x}_3\}$. We then resolve the second and third clause and get a saturation of ϕ w.r.t. x_1: $\{x_2, \overline{x}_2 \vee x_3, \overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3, x_1 \vee x_2 \vee x_3, \overline{x}_3\}$. From now on, we only consider the clauses not containing x_1: $C_1 = \{x_2, \overline{x}_2 \vee x_3, \overline{x}_3\}$, and ignore the clauses containing x_1: $\{\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3, x_1 \vee x_2 \vee x_3\}$. We continue by resolving the first two clauses of C_1; we get $\{x_3, x_2 \vee \overline{x}_3, \overline{x}_3\}$, which is a saturation of C_1 w.r.t. x_2. Hence, $C_2 = \{x_3, \overline{x}_3\}$ is the resulting multiset of clauses not containing x_2 and ignore $\{x_2 \vee \overline{x}_3\}$, which is the multiset of clauses containing x_2. Finally, we resolve $\{x_3, \overline{x}_3\}$ and get the empty clause. Since all the variables have been saturated, the minimum number of unsatisfied clauses in ϕ is 1.

The use of restrictions of MaxSAT resolution has not been limited to branch-and-bound solvers. Narodytska and Bacchus [NB14] used MaxSAT resolution in SAT-based MaxSAT solvers, avoiding the use of cardinality constraints and obtaining very competitive results on industrial instances.

There exists no polynomial-size resolution proof of the pigeon hole principle (PHP). However, Ignatiev et al. [IMM17] showed that there exist polynomial-size MaxSAT resolution proofs of PHP if PHP is encoded as a Partial MaxSAT instance using the dual rail encoding. Indeed, the combination of the dual rail encoding and MaxSAT resolution is a stronger proof system than either general resolution or conflict-driven clause learning [BBI+18].

MaxSAT resolution has been extended to the multiple-valued clausal forms known as signed CNF formulas [BHM00]. The defined signed MaxSAT resolution rules are complete and provide a logical framework for weighted constraint satisfaction problems (WCSP) [ABLM07b]. Besides, some restrictions of the rules enforce the defined local consistency properties for WCSPs in a natural way [ABLM07a, ABLM13].

23.4.2. Clause MaxSAT Tableaux

The clause SAT tableau calculus [D'A99, Häh01], which is is unsound for MaxSAT, was reformulated to become a sound and complete MaxSAT calculus [LMS16b]. Roughly speaking, the main differences are that the extension rule is applied in a branch until all its clauses have been expanded regardless of the contradictions already detected in that branch, and that clauses and literals cannot be used more than once in a branch.

A clause MaxSAT tableau for a multiset of clauses $\phi = \{C_1, \ldots, C_m\}$ is a tree with a finite number of branches whose nodes are labelled with clauses that are declared to be either active or inactive in each branch. It is constructed by a sequence of applications of the following rules:

Initialize: A tree with a single branch with m nodes such that each node is labelled with a clause of ϕ is a clause MaxSAT tableau for ϕ. Such a tableau is called initial tableau and its clauses are declared to be active.

Extension: Given a clause MaxSAT tableau T for ϕ, a branch B of T, and a node of B labelled with an active clause $l_1 \vee \cdots \vee l_r$ with $r \geq 2$, the tableau obtained by creating r sibling nodes below B and labelling each node with a different unit clause from $\{l_1, \ldots, l_r\}$ is a clause MaxSAT tableau for ϕ. Clause $l_1 \vee \cdots \vee l_r$ becomes inactive in the new branches, and unit clauses $l_1, \ldots, l_r$ are declared to be active.

Contradiction: Given a clause MaxSAT tableau T for ϕ, a branch B of T, and two nodes of B labelled with two active unit clauses l and $\bar{l}$, the tableau obtained by appending a node labelled with an empty clause below B is a clause MaxSAT tableau for ϕ. The empty clause becomes active and unit clauses l and $\bar{l}$ become inactive.

A clause MaxSAT tableau is completed when all its branches are saturated, in the sense that all its active clauses are empty and unit clauses, and the contradiction rule cannot be further applied in any branch. The cost of a branch in a completed tableau T is the number of empty clauses in it, and the cost of T is the minimum cost among all its branches.

The soundness and completeness of the clause MaxSAT tableau calculus states that the minimum number of clauses that can be unsatisfied in a multiset of clauses ϕ is k iff the cost of a completed clause MaxSAT tableau for ϕ

is k. Thus, the systematic construction of a completed clause MaxSAT tableau for ϕ provides an exact method for MaxSAT, and each completed tableau is a proof.

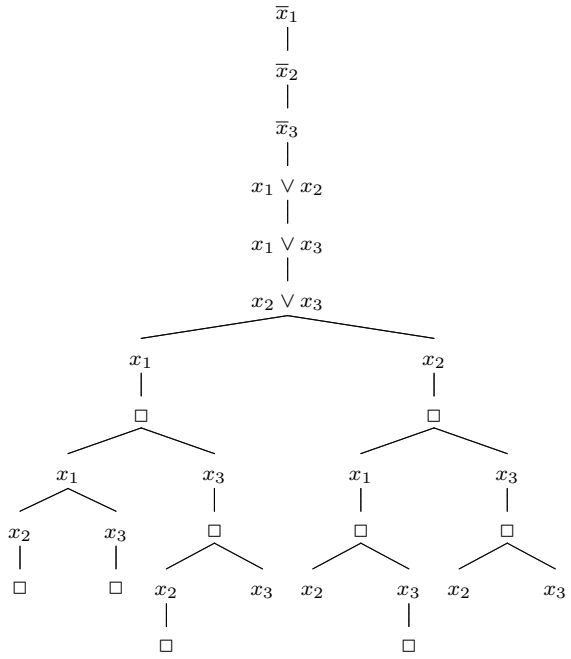

Figure 23.2. A completed clause MaxSAT tableau for $\phi = \{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$.

Figure 23.2 shows a completed clause MaxSAT tableau T for the multiset of clauses $\phi = \{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$. The saturated branches of the tableau have cost 2 except for branches 3 and 6 (counting from left to right) that have cost 3. The active clauses in each branch are: $\{\square, \square, x_1, \overline{x}_3\}$ (branch 1), $\{\square, \square, x_1, \overline{x}_2\}$ (branch 2), $\{\square, \square, \square\}$ (branch 3), $\{\square, \square, \overline{x}_2, x_3\}$ (branch 4), $\{\square, \square, x_2, \overline{x}_3\}$ (branch 5), $\{\square, \square, \square\}$ (branch 6), $\{\square, \square, \overline{x}_1, x_2\}$ (branch 7), and $\{\square, \square, \overline{x}_1, x_3\}$ (branch 8). Therefore, the minimum number of unsatisfied clauses in ϕ is 2.

Inspired on the previous calculus, as well as in the clause MinSAT calculus defined in [LMS16a], Argelich et al. [ALMS18] defined a proof procedure that implements a sound and complete clause tableau-style calculus for both MaxSAT and MinSAT. The expansion rules preserves adequately the number of unsatisfied clauses in the generated subproblems. The leaf nodes of a completed tableau contain a number of empty clauses ranging between the minimum and the maximum number of unsatisfied clauses in the input formula, and there is at least one branch with the minimum value and at least one branch with the maximum value.

Casas-Roma et al. [CHM17] defined a complete natural deduction calculus for MaxSAT which was inspired on the previous clause MaxSAT tableau calculus.

23.5. Approximation Algorithms

Heuristic local search algorithms [1] are often quite effective at finding near-optimal solutions. Actually, most of the exact MaxSAT solvers use a local search algorithm to compute an initial upper bound. However, these algorithms, in contrast with approximation algorithms, do not come with rigorous guarantees concerning the quality of the final solution or the required maximum runtime. Informally, an algorithm approximately solves an optimization problem if it always returns a feasible solution whose measure is close to optimal, for example, within a factor bounded by a constant or by a slowly growing function of the input size. Given a constant α, an algorithm is an α-approximation algorithm for a maximization (minimization) problem if it provides a feasible solution in polynomial time which is at least (most) α times the optimum, considering all the possible instances of the problem.

The first MaxSAT approximation algorithm, with a performance guarantee of $\frac{1}{2}$, is a greedy algorithm that was devised by Johnson in 1974 [Joh74]. This result was improved in 1994 by Yannakakis [Yan94], and Goemans and Williamson [GW94b], who described 3/4–approximation algorithms for MaxSAT. Then, Goemans and Williamson [GW94b] proposed a .878-approximation algorithm for Max2SAT (which gives a .7584-approximation for MaxSAT [GW95]) based on semidefinite programming [GW94a]. Since then other improvements have been achieved, but there is a limit on approximability: Hastad [Has97] proved that, unless $P = NP$, no approximation algorithm for MaxSAT (even for Max3SAT) can achieve a performance guarantee better than 7/8. Interestingly, Karloff and Zwick [KZ97] gave a 7/8 approximation algorithm for Max3SAT, showing that the constant 7/8 is tight. The most promising approaches from a theoretical and practical point of view are based on semidefinite programming [GvHL06]. We refer the reader to the survey of Anjos [Anj05] to learn more about how to approximate MaxSAT with semidefinite programming.

23.6. The MaxSAT Evaluation

The MaxSAT Evaluation [ALMP08, ALMP11a, ALMP11b] is an affiliated event of the International Conference on Theory and Applications of Satisfiability Testing. It has been held annually since 2006 and has been decisive for promoting and advancing MaxSAT solving.

The main goals of the MaxSAT Evaluation are to assess the state of the art in the field of MaxSAT solvers, collect and re-distribute a heterogeneous MaxSAT benchmark set for further scientific evaluations, and promote MaxSAT as a viable option for solving instances of a wide range of NP-hard optimization problems. In the beginning, only exact solvers were evaluated. Ultimately, an evaluation of non-exact solvers is also conducted.

Until 2016, the MaxSAT Evaluation had the random, crafted and industrial categories and the unweighted MaxSAT, weighted MaxSAT, partial MaxSAT and weighted partial MaxSAT tracks. Since 2017, it has two tracks: unweighted and

[1] We do not include a section about local search and MaxSAT because there is a chapter on local search in the handbook.

weighted. The unweighted track combines the industrial and crafted unweighted and unweighted partial MaxSAT categories from previous MaxSAT evaluations. Purely randomly generated instances are not included. The weighted track combines the industrial and crafted weighted and weighted partial MaxSAT categories from previous MaxSAT evaluations. All benchmarks contain soft clauses with different weights. Purely randomly generated instances are not included.

We refer the reader to https://maxsat-evaluations.github.io/2018/ for additional information and the results of the last edition of the MaxSAT Evaluation.

23.7. Other Contributions to MaxSAT

First of all, we should mention a variety of applications of MaxSAT in a range of real-world domains as diverse as bioinformatics [GL12, MAGL11], circuit design and debugging [SMV+07], community detection in complex networks [JMRS17], diagnosis [DG12], FPGA routing [XRS03], planning [ZB12], scheduling [BGSV15], team formation [MNRS17] and time tabling [AN14], among many others.

Other research topics that have appeared in the literature and contain substantial contributions to the field of MaxSAT solving include the definition of clause clause learning schemes in branch-and-bound solvers [AH16, AM06b] that are not yet competitive enough to solve industrial instances, the creation of robust MaxSAT solutions [BBMV13], the definition of efficient encodings from MaxCSP to MaxSAT [ACLM12], the extension of MaxSAT to many-valued logic [ALM17] and the definition of MaxSAT formalisms that deal with blocks of soft clauses instead of individual soft clauses [AM06a, HMM15].

23.8. The MinSAT Problem

Given the success of MaxSAT, the community has started to look into MinSAT. At first sight, one could think that the solving techniques and encodings to be used in MinSAT are very similar to the ones used in MaxSAT and, therefore, that there is no need of investigating MinSAT from a problem solving perspective. However, most of the research conducted so far indicates that they may be quite different, as well as that the performance profile of MaxSAT and MinSAT is also different for several optimization problems represented into these formalisms [ALMZ14, IMM14, LZMS12]. It is also worth mentioning that MinSAT is meaningful for both satisfiable and unsatisfiable instances, whereas MaxSAT is only meaningful for unsatisfiable instances. A closely related problem has been analyzed in [IMPMS13, IMPM16].

The work on MinSAT for solving optimization problems may be divided into the following categories:

- Transformations between MinSAT and MaxSAT: Reductions from MinSAT to partial MaxSAT were defined in [LMQZ10], but these reductions do not generalize to weighted partial MinSAT. This drawback was overcome with the definition of the natural encoding [KÏ2], which was improved in [ZLMA12]. Reductions of weighted partial MinSAT to Group MaxSAT were evaluated in [HMPMS12].

- Encodings from weighted MaxCSP to MinSAT: Efficient encodings were defined in [ALMZ13]. Using the MaxSAT direct encoding [ACLM12], we must add one clause for every no-good, while using the MinSAT direct encoding [ALMZ13], we must instead add one clause for every good. This implies, for instance, that for representing the constraint $X = Y$, we need a number of clauses linear in the domain size in MinSAT, and a quadratic number of clauses in MaxSAT. We are in the opposite situation if we want to represent the constraint $X \neq Y$, So, it seems that MaxSAT and MinSAT could be complementary in some scenarios [ALMZ13].
- Complete logical calculi: MaxSAT resolution is sound for MinSAT but to get completeness the elimination of variables must be defined differently [LM15]. After saturating a variable x, the clauses containing the variable x are ignored in MaxSAT but, in MinSAT, the resulting clauses of eliminating the occurrences of both x and $\overline{x}$ must also be considered in the saturation of the next variable. In this way, after saturating all the variables, the number of empty clauses derived is equal to the maximum number of clauses that can be unsatisfied. In the case of clause tableaux, the clause MaxSAT calculus is unsound for MinSAT because the extension rule does not preserve the maximum number of unsatisfied clauses. A sound and complete clause MinSAT calculus was defined in [LMS16a].
- MinSAT solvers: The only existing branch-and-bound MinSAT solver, MinSatz [LZMS11, LZMS12], is based on MaxSatz and implements upper bounds that exploit clique partition algorithms and MaxSAT technology. There exist two SAT-based MinSAT solvers of this class [ALMZ12, HMPMS12]. They differ with SAT-based MaxSAT solvers in the way of relaxing soft clauses. A local search MinSAT solver was described in [AH15a].

23.9. Conclusions

We have presented an overview about MaxSAT and focused on the solving techniques that have proved to be useful in terms of performance. When we wrote the chapter on MaxSAT for the first edition of this handbook, MaxSAT solving was still an incipient research topic but nowadays can be considered a mature research topic with a variety of applications and also of open problems.

From the perspective of branch-and-bound MaxSAT solvers, the main challenge is to build a solver as competitive as SAT-based MaxSAT solvers on industrial instances. A first step in this direction could be the definition of a powerful clause learning mechanism for soft clauses. From a logical perspective, an interesting research avenue is the study of the proof complexity of logical calculi for MaxSAT to find proof systems more efficient that the proof systems used in SAT solving. From a knowledge representation perspective, an interesting research path is to find clever encodings of combinatorial optimization problems and understand the impact of modeling on the performance of MaxSAT solvers, as well as the definition of richer formalisms like non-clausal MaxSAT.

Acknowledgments

This work was supported by Project LOGISTAR from the EU H2020 Research and Innovation Programme under Grant Agreement No. 769142, and MINECO-FEDER project RASO (TIN2015-71799-C2-1-P).

References

[ABLM07a] C. Ansótegui, M. L. Bonet, J. Levy, and F. Manyà. Inference rules for high-order consistency in weighted CSP. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, Canada*, pages 167–172, 2007.

[ABLM07b] C. Ansótegui, M. L. Bonet, J. Levy, and F. Manyà. The logic behind weighted CSP. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, Hyderabad, India*, pages 32–37, 2007.

[ABLM13] C. Ansótegui, M. L. Bonet, J. Levy, and F. Manyà. Resolution procedures for multiple-valued optimization. *Information Sciences*, 227:43–59, 2013.

[ACLM12] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Efficient encodings from CSP into SAT, and from MaxCSP into MaxSAT. *Multiple-Valued Logic and Soft Computing*, 19(1-3):3–23, 2012.

[AG13] C. Ansótegui and J. Gabàs. Solving (weighted) partial MaxSAT with ILP. In *Proceedings of the 10th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2013, Yorktown Heights, NY, USA*, pages 403–409, 2013.

[AH14a] A. Abramé and D. Habet. Ahmaxsat: Description and evaluation of a branch and bound Max-SAT solver. *JSAT*, 9:89–128, 2014.

[AH14b] A. Abramé and D. Habet. Local max-resolution in branch and bound solvers for Max-SAT. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI, Limassol, Cyprus*, pages 336–343, 2014.

[AH14c] A. Abramé and D. Habet. Maintaining and handling all unit propagation reasons in exact Max-SAT solvers. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS, Prague, Czech Republic*, 2014.

[AH15a] A. Abramé and D. Habet. Local search algorithm for the partial minimum satisfiability problem. In *Proceedings of the 27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI, Vietri sul Mare, Italy*, pages 821–827, 2015.

[AH15b] A. Abramé and D. Habet. On the resiliency of unit propagation to Max-Resolution. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI, Buenos Aires, Argentina*, pages 268–274, 2015.

[AH16] A. Abramé and D. Habet. Learning nobetter clauses in Max-SAT branch and bound solvers. In *Proceedings of the 28th IEEE Interna-*

tional Conference on Tools with Artificial Intelligence, ICTAI, San Jose, CA, USA, pages 452–459, 2016.

[ALM17] J. Argelich, C. M. Li, and F. Manyà. Exploiting many-valued variables in MaxSAT. In *Proceedings of the 7th IEEE International Symposium on Multiple-Valued Logic, ISMVL, Novi Sad, Serbia*, pages 155–160, 2017.

[ALMP08] J. Argelich, C. M. Li, F. Manyà, and J. Planes. The first and second Max-SAT evaluations. *Journal on Satisfiability, Boolean Modeling and Computation*, 4, 2008.

[ALMP11a] J. Argelich, C. M. Li, F. Manyà, and J. Planes. Analyzing the instances of the maxsat evaluation. In *Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing, SAT-2011, Ann Arbor, MI, USA*, pages 360–361. Springer LNCS 6695, 2011.

[ALMP11b] J. Argelich, C. M. Li, F. Manyà, and J. Planes. Experimenting with the instances of the MaxSAT evaluation. In *Proceedings of the 14th International Conference of the Catalan Association for Artificial Intelligence, CCIA-2011, Lleida, Spain*, pages 31–40. IOS Press, 2011.

[ALMS18] J. Argelich, C. M. Li, F. Manyà, and J. R. Soler. Clause branching in MaxSAT and MinSAT. In *Proceedings of the 21st International Conference of the Catalan Association for Artificial Intelligence, Roses, Spain*, volume 308 of *Frontiers in Artificial Intelligence and Applications*, pages 17–26. IOS Press, 2018.

[ALMZ12] C. Ansótegui, C. M. Li, F. Manyà, and Z. Zhu. A SAT-based approach to MinSAT. In *Proceedings of the 15th International Conference of the Catalan Association for Artificial Intelligence, CCIA-2012, Alacant, Spain*, pages 185–189. IOS Press, 2012.

[ALMZ13] J. Argelich, C. M. Li, F. Manyà, and Z. Zhu. MinSAT versus MaxSAT for optimization problems. In *Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming, CP 2013, Uppsala, Sweden*, pages 133–142. Springer LNCS 8124, 2013.

[ALMZ14] J. Argelich, C. M. Li, F. Manyà, and Z. Zhu. Many-valued MinSAT solving. In *Proceedings, 44th International Symposium on Multiple-Valued Logics (ISMVL), Bremen, Germany*, pages 32–37. IEEE CS Press, 2014.

[AM06a] J. Argelich and F. Manyà. Exact Max-SAT solvers for over-constrained problems. *Journal of Heuristics*, 12(4–5):375–392, 2006.

[AM06b] J. Argelich and F. Manyà. Learning hard constraints in Max-SAT. In *Proceedings of the Workshop on Constraint Solving and Constraint Logic Porgramming, CSCLP-2006, Lisbon, Portugal*, pages 1–12, 2006.

[AMP03] T. Alsinet, F. Manyà, and J. Planes. Improved branch and bound algorithms for Max-SAT. In *Proceedings of the 6th International Conference on the Theory and Applications of Satisfiability Testing*, 2003.

[AMP04] T. Alsinet, F. Manyà, and J. Planes. A Max-SAT solver with lazy data structures. In *Proceedings of the 9th Ibero-American Conference on Artificial Intelligence, IBERAMIA 2004, Puebla, México*, pages 334–342. Springer LNCS 3315, 2004.

[AMP05] T. Alsinet, F. Manyà, and J. Planes. Improved exact solver for weighted Max-SAT. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing, SAT-2005, St. Andrews, Scotland*, pages 371–377. Springer LNCS 3569, 2005.

[AMP08] T. Alsinet, F. Manyà, and J. Planes. An efficient solver for Weighted Max-SAT. *Journal of Global Optimization*, 41:61–73, 2008.

[AN14] R. J. A. Achá and R. Nieuwenhuis. Curriculum-based course timetabling with SAT and MaxSAT. *Annals OR*, 218(1):71–91, 2014.

[Anj05] M. F. Anjos. Semidefinite optimization approaches for satisfiability and maximum-satisfiability problems. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:1–47, 2005.

[BBI+18] M. L. Bonet, S. Buss, A. Ignatiev, J. Marques-Silva, and A. Morgado. MaxSAT resolution with the dual rail encoding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI, New Orleans, Louisiana, USA*, pages 6565–6572, 2018.

[BBMV13] M. Bofill, D. Busquets, V. Muñoz, and M. Villaret. Reformulation based MaxSAT robustness. *Constraints*, 18(2):202–235, 2013.

[BF99] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2:299–306, 1999.

[BGSV15] M. Bofill, M. Garcia, J. Suy, and M. Villaret. MaxSAT-based scheduling of B2B meetings. In *Proceedings of the12th International Conference on Integration of AI and OR Techniques in Constraint Programming, CPAIOR, Barcelona, Spain*, pages 65–73, 2015.

[BHM00] B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In D. Basin, M. D'Agostino, D. Gabbay, S. Matthews, and L. Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht, 2000.

[BLM06] M. L. Bonet, J. Levy, and F. Manyà. A complete calculus for Max-SAT. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT-2006, Seattle, USA*, pages 240–251. Springer LNCS 4121, 2006.

[BLM07] M. L. Bonet, J. Levy, and F. Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8–9):240–251, 2007.

[BR99] N. Bansal and V. Raman. Upper bounds for MaxSat: Further improved. In *Proc 10th International Symposium on Algorithms and Computation, ISAAC'99, Chennai, India*, pages 247–260. Springer, LNCS 1741, 1999.

[CHM17] J. Casas-Roma, A. Huertas, and F. Manyà. Solving MaxSAT with natural deduction. In *Proceedings of the 20th International Conference of the Catalan Association for Artificial Intelligence, Deltebre, Spain*, volume 300 of *Frontiers in Artificial Intelligence and Appli-*

cations, pages 186–195. IOS Press, 2017.

[D'A99] M. D'Agostino. Tableaux methods for classical propositional logic. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 45–123. Kluwer, 1999.

[DDDL07] S. Darras, G. Dequen, L. Devendeville, and C. M. Li. On inconsistent clause-subsets for max-sat solving. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming, CP-2007, Providence/RI, USA*, pages 225–240. Springer LNCS 4741, 2007.

[DG12] D. D'Almeida and É. Grégoire. Model-based diagnosis with default information implemented through MAX-SAT technology. In *Proceedings of the IEEE 13th International Conference on Information Reuse & Integration, IRI, Las Vegas, NV, USA*, pages 33–36, 2012.

[dGLMS03] S. de Givry, J. Larrosa, P. Meseguer, and T. Schiex. Solving Max-SAT as weighted CSP. In *9th International Conference on Principles and Practice of Constraint Programming, CP-2003, Kinsale, Ireland*, pages 363–376. Springer LNCS 2833, 2003.

[GL12] J. Guerra and I. Lynce. Reasoning over biological networks using maximum satisfiability. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming, CP, Québec City, QC, Canada*, pages 941–956, 2012.

[GvHL06] C. P. Gomes, W.-J. van Hoeve, and L. Leahu. The power of semidefinite programming relaxations for MAXSAT. In *Proceedings of the Third International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR06, Cork, Ireland*, pages 104–118. Springer LNCS 3990, 2006.

[GW94a] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 422–431, 1994.

[GW94b] M. X. Goemans and D. P. Williamson. New 3/4–approximation algorithms for the maximum satisfiability problem. *SIAM Journal of Discrete Mathematics*, 7:656–666, 1994.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[Häh01] R. Hähnle. Tableaux and related methods. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 100–178. Elsevier and MIT Press, 2001.

[Has97] J. Hastad. Some optimal inapproximability results. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*, pages 1–10, 1997.

[HL06] F. Heras and J. Larrosa. New inference rules for efficient Max-SAT solving. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-2006, Boston/MA, USA*, pages 68–73, 2006.

[HLO07] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: A new weighted Max-SAT solver. In *Proceedings of the 10th International Confer-*

ence on Theory and Applications of Satisfiability Testing, SAT-2007, Lisbon, Portugal, pages 41–55, 2007.

[HLO08] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[HMM15] F. Heras, A. Morgado, and J. Marques-Silva. MaxSAT-based encodings for group MaxSAT. *AI Communications*, 28(2):195–214, 2015.

[HMPMS12] F. Heras, A. Morgado, J. Planes, and J. Marques-Silva. Iterative SAT solving for minimum satisfiability. In *Proceedings of the IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece*, pages 922–927, 2012.

[IMM14] A. Ignatiev, A. Morgado, and J. Marques-Silva. On reducing maximum independent set to minimum satisfiability. In *Proc 17th International Conference on Theory and Applications of Satisfiability Testing, SAT, Vienna, Austria*, pages 103–120. Springer, LNAI 8561, 2014.

[IMM17] A. Ignatiev, A. Morgado, and J. Marques-Silva. On tackling the limits of resolution in SAT solving. In *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing, SAT, Melbourne, Australia*, pages 164–183. Springer LNCS 10491, 2017.

[IMPM16] A. Ignatiev, A. Morgado, J. Planes, and J. Marques-Silva. Maximal falsifiability. *AI Communications*, 29(2):351–370, 2016.

[IMPMS13] A. Ignatiev, A. Morgado, J. Planes, and J. Marques-Silva. Maximal falsifiability - definitions, algorithms, and applications. In *Proc 19th International Conference on Logic Programming and Automated Theorem Proving, LPAR, Stellenbosch, South Africa*, pages 439–456. Springer, LNAI 8312, 2013.

[JLM17] H. Jiang, C. M. Li, and F. Manyà. An exact algorithm for the maximum weight clique problem of large graphs. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence, AAAI, San Francisco/CA, USA*, 2017.

[JMRS17] S. Jabbour, N. Mhadhbi, B. Raddaoui, and L. Sais. A SAT-based framework for overlapping community detection in networks. In *Proceedings of the 21st Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, Part II, PAKDD, Jeju, South Korea*, pages 786–798, 2017.

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[KÏ2] A. Kügel. Natural Max-SAT encoding of Min-SAT. In *Proceedings of the Learning and Intelligent Optimization Conference, LION 6, Paris, France*, 2012.

[Kue10] A. Kuegel. Improved exact solver for the Weighted MAX-SAT problem. In *Proceedings of Workshop Pragmatics of SAT, POS-10, Edinburgh, UK*, pages 15–27, 2010.

[KZ97] H. Karloff and U. Zwick. A 7/8-approximation algorithm for

MAX3SAT ? In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, FOCS'97*, pages 406–415, 1997.

[LFX13] C. Li, Z. Fang, and K. Xu. Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem. In *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI, Herndon, VA, USA*, pages 939–946, 2013.

[LH05] J. Larrosa and F. Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-2005, Edinburgh, Scotland*, pages 193–198. Morgan Kaufmann, 2005.

[LHdG08] J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2–3):204–233, 2008.

[LJM17] C. Li, H. Jiang, and F. Manyà. On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem. *Computers & OR*, 84:1–15, 2017.

[LJX15] C. Li, H. Jiang, and R. Xu. Incremental maxsat reasoning to reduce branches in a branch-and-bound algorithm for maxclique. In *Proceedings of the 9th International Conference on Learning and Intelligent Optimization, LION, Lille, France*, pages 268–274, 2015.

[LM15] C. M. Li and F. Manyà. An exact inference scheme for MinSAT. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI-2015, Buenos Aires, Argentina*, pages 1959–1965, 2015.

[LMMP08] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Transforming inconsistent subformulas in MaxSAT lower bound computation. In *Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming, CP-2008, Sydney, Australia.* Springer LNCS, 2008.

[LMMP09] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Exploiting cycle structures in Max-SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK*, pages 467–480. Springer LNCS 5584, 2009.

[LMMP10] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Resolution-based lower bounds in MaxSAT. *Constraints*, 15(4):456–484, 2010.

[LMP05] C. M. Li, F. Manyà, and J. Planes. Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, CP-2005, Sitges, Spain*, pages 403–414. Springer LNCS 3709, 2005.

[LMP06] C. M. Li, F. Manyà, and J. Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proceedings of the 21st National Conference on Artificial Intelligence, AAAI-2006, Boston/MA, USA*, pages 86–91, 2006.

[LMP07] C. M. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.

[LMQZ10] C. M. Li, F. Manyà, Z. Quan, and Z. Zhu. Exact MinSAT solving. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing, SAT-2010, Edinburgh, UK*, pages 363–368. Springer LNCS 6175, 2010.

[LMS16a] C. M. Li, F. Manyà, and J. R. Soler. A clause tableau calculus for MinSAT. In *Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2016, Barcelona, Spain*, volume 288 of *Frontiers in Artificial Intelligence and Applications*, pages 88–97. IOS Press, 2016.

[LMS16b] C. M. Li, F. Manyà, and J. R. Soler. A clause tableaux calculus for MaxSAT. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI-2016, New York, USA*, pages 766–772, 2016.

[LQ10] C. M. Li and Z. Quan. An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI, Atlanta, Georgia, USA*, 2010.

[LS07] H. Lin and K. Su. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, Hyderabad, India*, pages 2334–2339, 2007.

[LSL08] H. Lin, K. Su, and C. M. Li. Within-problem learning for efficient lower bound computation in Max-SAT solving. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI-2008, Chicago/IL, USA*, pages 351–356, 2008.

[LZMS11] C. M. Li, Z. Zhu, F. Manyà, and L. Simon. Minimum satisfiability and its applications. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, Barcelona, Spain*, pages 605–610, 2011.

[LZMS12] C. M. Li, Z. Zhu, F. Manyà, and L. Simon. Optimizing with minimum satisfiability. *Artificial Intelligence*, 190:32–44, 2012.

[MAGL11] J. Marques-Silva, J. Argelich, A. Graça, and I. Lynce. Boolean lexicographic optimization: algorithms & applications. *Annals of Matematics and Artificial Intelligence*, 62(3-4):317–343, 2011.

[MNRS17] F. Manyà, S. Negrete, C. Roig, and J. R. Soler. A MaxSAT-based approach to the team composition problem in a classroom. In *Autonomous Agents and Multiagent Systems - AAMAS 2017 Workshops, Visionary Papers, São Paulo, Brazil, Revised Selected Papers*, pages 164–173. Springer LNCS 10643, 2017.

[NB14] N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence, Québec City, Canada.*, pages 2717–2723, 2014.

[NR00] R. Niedermeier and P. Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, 36:63–88, 2000.

[PD07] K. Pipatsrisawat and A. Darwiche. Clone: Solving weighted MaxSAT in a reduced search space. In *20th Australian Joint Conference*

on Artificial Intelligence, AI-07, Qeensland, Australia, pages 223–233, 2007.

[PPC+08] K. Pipatsrisawat, A. Palyan, M. Chavira, A. Choi, and A. Darwiche. Solving weighted Max-SAT problems in a reduced search space: A performance analysis. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:191–217, 2008.

[RG07] M. Ramírez and H. Geffner. Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming, CP-2007, Providence/RI, USA*, pages 605–619. Springer LNCS 4741, 2007.

[SMV+07] S. Safarpour, H. Mangassarian, A. G. Veneris, M. H. Liffiton, and K. A. Sakallah. Improved design debugging using maximum satisfiability. In *Proceedings of 7th International Conference on Formal Methods in Computer-Aided Design, FMCAD, Austin, Texas, USA*, pages 13–19, 2007.

[SZ04] H. Shen and H. Zhang. Study of lower bound functions for max-2-sat. In *Proceedings of the 19th National Conference on Artificial Intelligence, AAAI-2004, San Jose/CA, USA*, pages 185–190, 2004.

[WF96] R. J. Wallace and E. Freuder. Comparative studies of constraint satisfaction and Davis-Putnam algorithms for maximum satisfiability problems. In D. Johnson and M. Trick, editors, *Cliques, Coloring and Satisfiability*, volume 26, pages 587–615. American Mathematical Society, 1996.

[XRS03] H. Xu, R. A. Rutenbar, and K. A. Sakallah. sub-sat: a formulation for relaxed boolean satisfiability with applications in routing. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(6):814–820, 2003.

[XZ04] Z. Xing and W. Zhang. Efficient strategies for (weighted) maximum satisfiability. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming, CP-2004, Toronto, Canada*, pages 690–705. Springer, LNCS 3258, 2004.

[XZ05] Z. Xing and W. Zhang. An efficient exact algorithm for (weighted) maximum satisfiability. *Artificial Intelligence*, 164(2):47–80, 2005.

[Yan94] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

[ZB12] L. Zhang and F. Bacchus. MAXSAT heuristics for cost optimal planning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence, Toronto, Ontario, Canada*, pages 1846–1852, 2012.

[ZLMA12] Z. Zhu, C. M. Li, F. Manyà, and J. Argelich. A new encoding from MinSAT into MaxSAT. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming, CP 2012, Québec City, QC, Canada*, pages 455–463. Springer LNCS 7514, 2012.

[ZSM03] H. Zhang, H. Shen, and F. Manyà. Exact algorithms for MAX-SAT. *Electronic Notes in Theoretical Computer Science*, 86(1), 2003.